



PROJETO </> EDUCAÇÃO

DO FUTURO

MÓDULO 05

Node.js e Git

AULA 04



AULA 04

CONTEÚDOS

- Apresentação dos principais Core Modules
- http: módulo para criar servidores HTTP;
- path: extrair informações de paths (caminhos) de arquivos;
- fs: file system, leitura e escrita de arquivos;
- url: módulo para trabalhar com URLs;
- Exercícios



O módulo http

- Podemos criar um servidor HTTP utilizando o módulo http do Node.js. Esse servidor será responsável por receber uma requisição e enviar uma resposta com código HTML.
- Para isso, utilizamos alguns métodos importantes, como o `createServer`, que é usado para criar o servidor, e o `listen`, que define a porta em que o servidor ficará aguardando as requisições.



O módulo http

- Exemplo:

```
const http = require('http')
```

```
const port = 3000
```

```
const server = http.createServer((req, res) => {  
  res.write('Oi HTTP')  
  res.end()  
})
```

```
server.listen(port, () => {  
  console.log(`Servidor rodando na porta: ${port}`)  
})
```



Retornando HTML com o http

- Para retornar HTML, precisamos adicionar alguns recursos
- Incluir um status code no retorno, usando a propriedade `statusCode` Alterar os headers para `text/html`
- Retornar o HTML com o método `end` do `http`



Retornando HTML com o http

- Exemplo:

```
ando_projeto  JS index.js ...\5_pacote_global  JS index.js ...\1_http  JS index.js ...\2_r
curso_node-main > 3_CORE_MODULES > 2_retornando_html > JS index.js > [?] server > [?] http
1  const http = require('http');
2
3  const port = 3000;
4
5  const server = http.createServer((req, res) => {
6    res.statusCode = 200;
7    res.setHeader('Content-Type', 'text/html; charset=utf-8');
8    res.end('<h1>Olá, este é o meu primeiro server!</h1>');
9  });
10
11 server.listen(port, () => {
12   console.log(`Servidor rodando na porta: ${port}`);
13 });
```



Módulos url

- O módulo url serve para decompor uma URL que passamos para o método parse;
- Podemos resgatar: host, path, search, query e etc;
- A partir destas informações podemos alterar a lógica do nosso código;
- Vamos ver na prática!



Módulos url

- Exemplo:

JS index.js X

curso_node-main > 3_CORE_MODULES > 3_url > JS index.js > ...

```
1  const url = require('url')
2  const address = 'https://www.meusite.com.br/catalogo?produtos=cadeira'
3  const parsedUrl = new url.URL(address)
4
5  console.log(parsedUrl.host)
6  console.log(parsedUrl.pathname)
7  console.log(parsedUrl.search)
8  console.log(parsedUrl.searchParams)
9  console.log(parsedUrl.searchParams.get('produtos'))
```



Unindo os módulos http e url

- Podemos trabalhar com esses módulos juntos para obter um resultado interessante
- Com o http, criamos o servidor e alteramos a resposta com base na URL acessada
- Com o url, processamos os parâmetros que vêm pela query string para alterar a lógica do http
- Agora, vamos ver na prática!



Unindo os módulos http e url

- Exemplo:

```
index.js ...\1_http    JS index.js ...\2_retornando_html    JS index.js ...\3_url    JS index.js ...\4_ht
curso_node-main > 3_CORE_MODULES > 4_http_com_url > JS index.js > ...
 1  const http = require("http");
 2  const url = require("url");
 3  const port = 3000;
 4  const server = http.createServer((req, res) => {
 5    var urlInfo = url.parse(req.url, true);
 6    const name = urlInfo.query.name;
 7    console.log(name)
 8    res.statusCode = 200;
 9    res.setHeader("Content-Type", 'text/html; charset=utf-8');
10    if (!name) {
11      res.end(
12        "<h1>Preencha seu nome:</h1><form method='GET'><input type='text'
13          name='name' /><input type='submit' value='Enviar'></form>"
14      );
15    } else {
16      res.end(`<h1>Seja bem-vindo ${name}!</h1>`);
17    }
18  });
19  server.listen(port, () => {
20    console.log(`Servidor rodando na porta: ${port}`);
21  });
```



Módulo fs

- O módulo fs (File System) serve para trabalhar com arquivos e diretórios
- Este é também um Core Module
- Podemos ler e escrever em arquivos, por exemplo
- Uma utilização interessante: logs do sistema
- Agora, vamos ver na prática!



Módulo fs

- Exemplo:

JS index.js X

curso_node-main > 3_CORE_MODULES > 5_fs > JS index.js > ...

```
1  const http = require("http");
2  const fs = require("fs");
3  const port = 3000;
4  const server = http.createServer((req, res) => {
5    fs.readFile("mensagem.html", function (err, data) {
6      res.writeHead(200, { "Content-Type": "text/html" });
7      res.write(data);
8      return res.end();
9    });
10 });
11
12 server.listen(port, () => {
13   console.log(`Servidor rodando na porta: ${port}`);
14 });
15
```

<> mensagem.html X

curso_node-main > 3_CORE_MODULES > 5_fs > <> mensagem.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width,
6        initial-scale=1.0" />
7      <title>Mensagem</title>
8    </head>
9    <body>
10     <h1>Este arquivo foi lido pelo fs!</h1>
11   </body>
12 </html>
```

Escrevendo em arquivos

- Podemos criar e escrever em arquivos utilizando o método writeFile
- Esta escrita pode estar associada a um conjunto de operações
- Como o envio de informações de um usuário, por exemplo
- Vamos unir mais uma vez os módulos na prática!



Escrevendo em arquivos

- Exemplo:

```
fs.writeFile("arquivo.txt", name, function (err,
data) {
  res.writeHead(302, {
    Location: "/",
  });
  return res.end();
});
```



Atualizando um arquivo

- O writeFile substitui tudo que está em um arquivo
- E se quisermos atualizar?
- Para este fim, utilizamos o appendFile
- Ele tem a mesma utilização que o writeFile, mas nos permite unir conteúdo
- Vamos ver na prática!



Atualizando um arquivo

- Exemplo:

```
fs.appendFile("arquivo.txt", nameNewLine, function
(err, data) {
  res.writeHead(302, {
    Location: "/",
  });
  return res.end();
});
```



Removendo um arquivo

- Para remover um arquivo com o fs utilizamos o método unlink;
- Precisamos passar o arquivo como parâmetro;
- Temos a possibilidade de checar se houve algum erro, a partir da callback retornada;
- Vamos ver na prática!



Removendo um arquivo

- Exemplo

```
JS index.js X
curso_node-main > 3_CORE_MODULES > 8_removendo_arquivos >
1  const fs = require('fs')
2
3  fs.unlink('arquivo.txt', function (err) {
4    if (err) {
5      console.log(err)
6      return
7    }
8    console.log('Arquivo removido!')
9  })
```



Renomeando um arquivo

- Para renomear um arquivo com o fs, usamos o método rename
- É necessário fornecer o nome atual do arquivo e o novo nome como parâmetros
- Se ocorrer algum erro, podemos verificar através da call-back
- Esse erro pode acontecer, por exemplo, se o arquivo não existir
- Agora, vamos ver como funciona na prática!



Renomeando um arquivo

- Exemplo:

```
JS index.js X
curso_node-main > 3_CORE_MODULES > 9_renameando_arquivo > JS index.js
1  const fs = require('fs')
2
3  fs.rename('arquivo.txt', 'novoarquivo.txt', function
  (err) {
4    if (err) {
5      console.log(err)
6      return
7    }
8    console.log('Arquivo renomeado!')
9  })
```



Detalhes de arquivos

- Podemos obter mais informações sobre os arquivos que temos acesso
- Utilizamos o método stat do fs
- Com ele, conseguimos saber: tamanho, data de criação, se é arquivo ou diretório, entre outros
- Agora, vamos ver na prática!



Detalhes de arquivos

- Exemplos

```
JS index.js  X
curso_node-main > 3_CORE_MODULES > 11_detalhes_arquivos > JS index.js >
1  const fs = require('fs')
2
3  fs.stat('novoarquivo.txt', (err, stats) => {
4    if (err) {
5      console.error(err)
6      return
7    }
8    console.log(stats.isFile())
9    console.log(stats.isDirectory())
10   console.log(stats.isSymbolicLink())
11   console.log(stats.ctime)
12   console.log(stats.size)
13 })
```



Módulo path

- Com o path, conseguimos extrair diversas informações sobre caminhos e arquivos
- Este também é um Core Module
- Algumas informações possíveis são: nome do diretório, nome do arquivo, extensão do arquivo, entre outras
- Agora, vamos ver na prática!



Módulo path

- Exemplo:

```
JS index.js  X
curso_node-main > 3_CORE_MODULES > 12_path > JS index.js > ...
1  const path = require('path')
2
3  const customPath = '/relatorios/Joao/relatorio1.pdf'
4
5  console.log(path.dirname(customPath))
6  console.log(path.basename(customPath))
7  console.log(path.extname(customPath))
```



Path absoluto e formar path

- Com a função resolve é possível saber qual o path completo até o arquivo alvo;
- E com a função join é possível formar um path dinâmico, com variáveis e valores fixos;
- São duas funções muito importantes;
- Vamos ver na prática!



Path absoluto e formar path

- Exemplo

```
JS index.js X
curso_node-main > 3_CORE_MODULES > 13_path_absoluto > JS index.j
1  const path = require('path')
2
3  // path absoluto
4  console.log(path.resolve('teste.txt'))
5
6  // formar path
7  const midFolder = 'relatorios'
8  const fileName = 'matheus.txt'
9
10 const finalPath = path.join('/', 'arquivos',
    midFolder, fileName)
11
12 console.log(finalPath)
```



Trabalhando com diretórios

- Com o módulo fs, também podemos trabalhar com diretórios (pastas)
- O método exists pode verificar se um diretório existe ou não
- E o método mkdir pode criar um diretório
- Agora, vamos ver na prática!



Trabalhando com diretórios

- Exemplo:

```
JS index.js X
curso_node-main > 3_CORE_MODULES > 14_trabalhando_com_dir
1  const fs = require('fs')
2
3  if (!fs.existsSync('./minhapasta')) {
4    | console.log('Não existe')
5  }
6
7  fs.mkdirSync('minhapasta')
8
9  if (fs.existsSync('minhapasta')) {
10 | console.log('Existe')
11 }
```



Módulo os

- Com o módulo os, podemos extrair informações do sistema operacional
- Este também é um Core Module
- Agora, vamos ver na prática!




Módulo os

- Exemplo

```
JS index.js X
curso_node-main > 3_CORE_MODULES > 15_n
1  const os = require("os");
2
3  console.log(os.cpus());
4
5  console.log(os.freemem());
6
7  console.log(os.homedir());
8
9  console.log(os.type());
```





(85) 98524-9935  youthspace

 contato@youthidiomas.com.br

<https://www.youthspace.com.br/>

2023

